REMARKS

INTRODUCTION:

In accordance with the foregoing, claims 5-7, 9, 11, 21 and 24- 27 have been amended, and claims 8 and 10 have been cancelled without prejudice or disclaimer. No new matter is being presented, and approval and entry are respectfully requested.

Claims 5-7, 9, 11, 21 and 24- 27 are pending and under consideration. Reconsideration is respectfully requested.

REJECTION UNDER 35 U.S.C. §112:

In the Office Action, at page 2-4, claims 5-11, 21, and 24-27 were rejected under 35 U.S.C. §112, second paragraph, for the reasons set forth therein. This rejection is traversed and reconsideration is requested.

It is respectfully submitted that the operations "generating, by a superposition calculating unit," and "calculating, by the superposition calculating unit" are intended to be separate calculations, wherein one calculates rmsd between (larger) point sets and the other is directed to calculating rmsd between elements of those point sets, i.e., between subsets of each larger set, to thus refine the combination of correspondence, as is illustrated in FIG. 13A-13D.

To clarify the above, claim 5 has been amended to recite, in part:

> generating, by a superposition calculating unit, a combination of correspondences comprising generating a combination based on minimized root mean square distance values between the first and second point sets, setting a predetermined threshold value and pruning a retrieval path if an attribute value of a candidate is greater than the predetermined threshold value, and determining that a point is a candidate if an attribute of an element of the first point set includes a type of an atom, an atomic group, a molecule, a hydrophilic property, a hydrophobic property, or a positive or negative charge that coincides with an attribute of an element of the first point set, and refining the elements of the first and second point sets based on an attribute of the elements of the first and second point sets to provide a first subset and a second subset, respectively;
>
> calculating, by the superposition calculating unit, a root mean square distance between the elements belonging to the first subset of the first point set relating to the elements belonging to the second subset of the second point set in the generated combination of correspondence.

Independent claims 21, 25, 26, and 27 have been amended similarly. Hence, the confusion concerning the operations of "generating, by a superposition calculating unit," and "calculating, by the superposition calculating unit" is now submitted to be resolved. Thus, amended independent claims 5, 21, 25, 26, and 27 are now submitted to be definite and in allowable form under 35 U.S.C. §112, second paragraph. Since claims 6-11 depend from amended independent claim 5, claims 6-11 are definite and in allowable form under

35 U.S.C. §112, second paragraph, for at least the reasons amended independent claim 5 is definite and in allowable form under 35 U.S.C. §112, second paragraph.

The Examiner seems confused about the meaning of the word "threshold." It is respectfully submitted that applicants are permitted to define terms in the specification. The terminology "threshold" is defined on page 35, line 31 through page 36, line 35, of the specification, wherein the terminology "threshold" value is recited as: "The point sets can be more efficiently related by setting a specified threshold value in the aforementioned methods (1) to (4) , and pruning a retrieval path if an attribute value of a candidate is greater than this threshold value. As this threshold value, for example, restriction in a nil number (the number of nil) and restriction in a r.m.s.d. value can be used." Restriction in a nil number and Restriction in an r.m.s.d. value are set forth on page 36, lines 4-35. FIG. 21 shows an example of pruning in a case where a total number of nil is restricted to 0 in relating a point set A = {a1, a2, a3} to a point set B = {b1, b2, b3}. IN FIG. 21, a portion designated at x in a tree structure is a portion to be pruned.

Hence, it is respectfully submitted that the terminology "threshold" of claims 5, 21, 25, 26 and 27 is clear to one skilled in the art and claims 5, 21, 25, 26 and 27 are definite and in allowable form under 35 U.S.C. §112, second paragraph, with respect to the threshold value.

The Examiner seems concerned about the terminology "pruning a retrieval path." Independent claim 5 has been amended to recite, in part: "generating, by a superposition calculating unit, a combination of correspondences comprising generating a combination by generating a decision tree having at least one retrieval path, the decision tree being based on minimized root mean square distance values between the first and second point sets, setting a predetermined threshold value and pruning a retrieval path if an attribute value of a candidate is greater than the predetermined threshold value, and determining that a point is a candidate if an attribute of an element of the first point set includes a type of an atom, an atomic group, a molecule, a hydrophilic property, a hydrophobic property, or a positive or negative charge that coincides with an attribute of an element of the first point set, and refining the elements of the first and second point sets based on an attribute of the elements of the first and second point sets to provide a first subset and a second subset, respectively."

Independent claims 21, 25, 26 and 27 have been amended in similar form.

Below, for the Examiner's convenience, is a description of how decision trees are generated, together with retrieval paths which may be pruned. It is respectfully submitted that the terminology "pruning a retrieval path" is known to those skilled in the art. For example, at http://dms.irb.hr/tutorial/tut_dtrees.php decision trees are described as follows:

Decision Trees

Decision trees are powerful and popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent *rules*. Rules can readily be expressed so that humans can understand them or even directly used in a database access language like SQL so that records falling into a particular category may be retrieved.

In some applications, the accuracy of a classification or prediction is the only thing that matters. In such situations we do not necessarily care how or why the model works. In other situations, the ability to explain the reason for a decision, is crucial. In marketing one has describe the customer segments to marketing professionals, so that they can utilize this knowledge in launching a successful marketing campaign. This domain experts must recognize and approve this discovered knowledge, and for this we need good descriptions. There are a variety of algorithms for building decision trees that share the desirable quality of interpretability. A well known and frequently used over the years is C4.5 (or improved, but commercial version See5/C5.0).

What is a decision tree ?

Decision tree is a classifier in the form of a tree structure (see Figure 1), where each node is either:

- a *leaf node* - indicates the value of the target attribute (class) of examples, or
- a *decision node* - specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test.

A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.

Decision tree induction is a typical inductive approach to learn knowledge on classification. The key requirements to do mining with decision trees are:

- *Attribute-value description*: object or case must be expressible in terms of a fixed collection of properties or attributes. This means that we need to discretize continuous attributes, or this must have been provided in the algorithm.
- *Predefined classes (target attribute values)*: The categories to which examples are to be assigned must have been established beforehand (supervised data).
- *Discrete classes*: A case does or does not belong to a particular class, and there

must be more cases than classes.

- *Sufficient data*: Usually hundreds or even thousands of training cases.

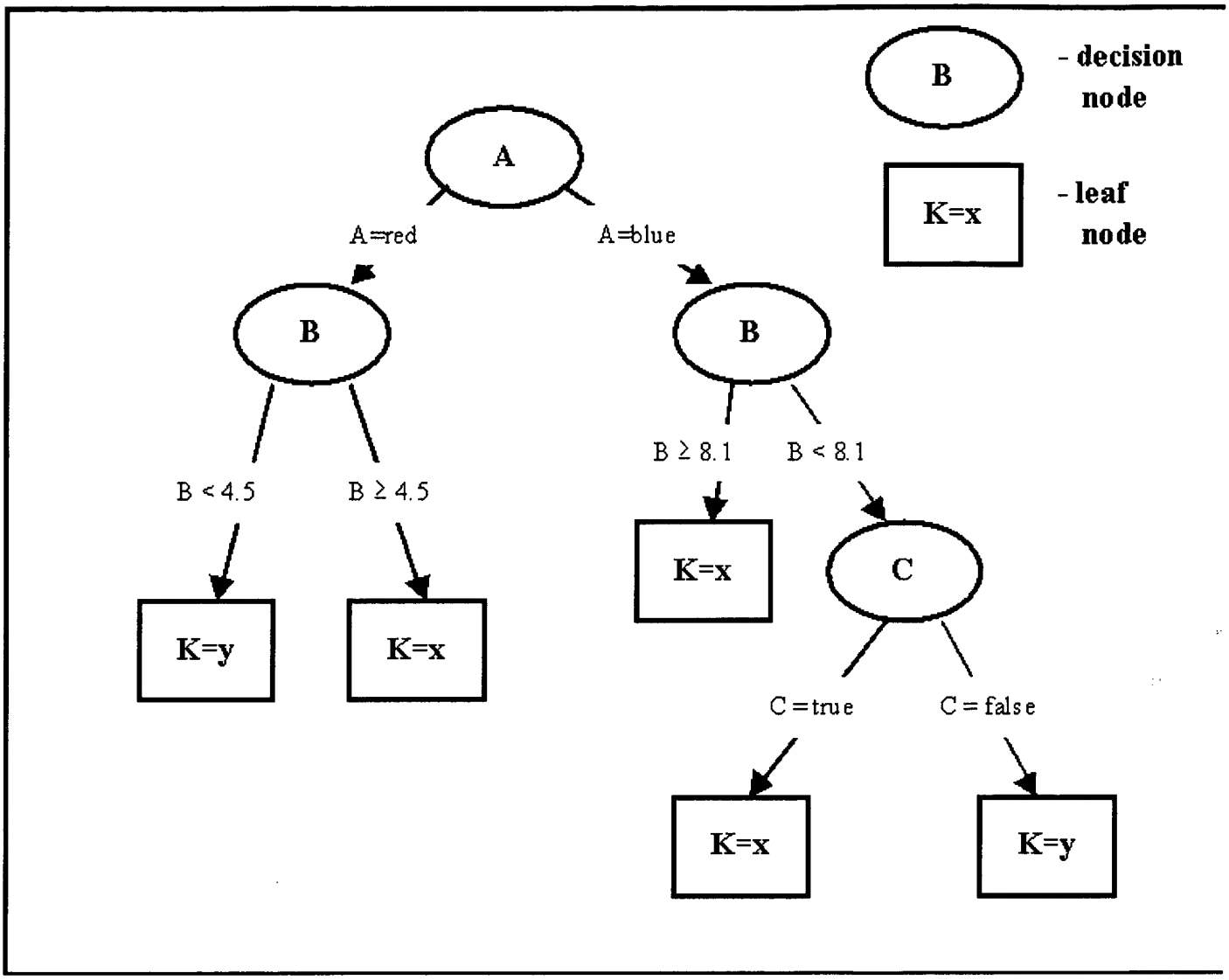

Figure 1: An example of a simple decision tree

Constructing decision trees

Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. Decision tree programs construct a decision tree *T* from a set of training cases.

J. Ross Quinlan originally developed ID3 at the University of Sydney. He first presented ID3 in 1975 in a book, *Machine Learning*, vol. 1, no. 1. ID3 is based on the Concept Learning System (CLS) algorithm.

---

function ID3

```
Input:  (R: a set of non-target attributes,
        C: the target attribute,
        S: a training set) returns a decision tree;
begin
  If S is empty, return a single node with
    value Failure;
  If S consists of records all with the same
    value for the target attribute,
    return a single leaf node with that value;
  If R is empty, then return a single node
    with the value of the most frequent of the
    values of the target attribute that are
    found in records of S; [in that case
    there may be errors, examples
    that will be improperly classified];
  Let A be the attribute with largest
    Gain(A,S) among attributes in R;
  Let {aj| j=1,2, .., m} be the values of
    attribute A;
  Let {Sj| j=1,2, .., m} be the subsets of
    S consisting respectively of records
    with value aj for A;
  Return a tree with root labeled A and arcs
    labeled a1, a2, .., am going respectively
    to the trees (ID3(R-{A}, C, S1), ID3(R-{A}, C, S2),
    .....,ID3(R-{A}, C, Sm);
  Recursively apply ID3 to subsets {Sj| j=1,2, .., m}
    until they are empty
end
```

Figure 2: ID3 Decision Tree Algorithm

ID3 searches through the attributes of the training instances and extracts the attribute that best separates the given examples. If the attribute perfectly classifies the training sets then ID3 stops; otherwise it recursively operates on the m (where m = number of possible values of an attribute) partitioned subsets to get their "best" attribute. The algorithm uses a greedy search, that is, it picks

the best attribute and never looks back to reconsider earlier choices. Note that ID3 may misclassify data.

The central focus of the decision tree growing algorithm is selecting which attribute to test at each node in the tree. For the selection of the attribute with the most inhomogeneous class distribution the algorithm uses the concept of entropy, which is explained next

Which attribute is the best classifier?

The estimation criterion in the decision tree algorithm is the selection of an attribute to test at each decision node in the tree. The goal is to select the attribute that is most useful for classifying examples. A good quantitative measure of the worth of an attribute is a statistical property called *information gain* that measures how well a given attribute separates the training examples according to their target classification. This measure is used to select among the candidate attributes at each step while growing the tree.

Entropy - a measure of homogeneity of the set of examples

In order to define information gain precisely, we need to define a measure commonly used in information theory, called entropy, that characterizes the (im)purity of an arbitrary collection of examples. Given a set S, containing only positive and negative examples of some target concept (a 2 class problem), the entropy of set S relative to this simple, binary classification is defined as:

$$\text{Entropy}(S) = - p_p \log_2 p_p - p_n \log_2 p_n$$

where p_p is the proportion of positive examples in S and p_n is the proportion of negative examples in S. In all calculations involving entropy we define 0log0 to be 0.

To illustrate, suppose S is a collection of 25 examples, including 15 positive and 10 negative examples [15+, 10-]. Then the entropy of S relative to this classification is

$$\text{Entropy}(S) = - (15/25) \log_2 (15/25) - (10/25) \log_2 (10/25) = 0.970$$

Notice that the entropy is 0 if all members of S belong to the same class. For example, if all members are positive ($p_p = 1$), then p_n is 0, and Entropy(S) = $-1' \log_2(1) - 0' \log_2 0 = -1' 0 - 0' \log_2 0 = 0$. Note the entropy is 1 (at its maximum!) when the collection contains an equal number of positive and negative examples. If the collection contains unequal numbers of positive and negative

examples, the entropy is between 0 and 1. Figure 3 shows the form of the entropy function relative to a binary classification, as p_+ varies between 0 and 1.

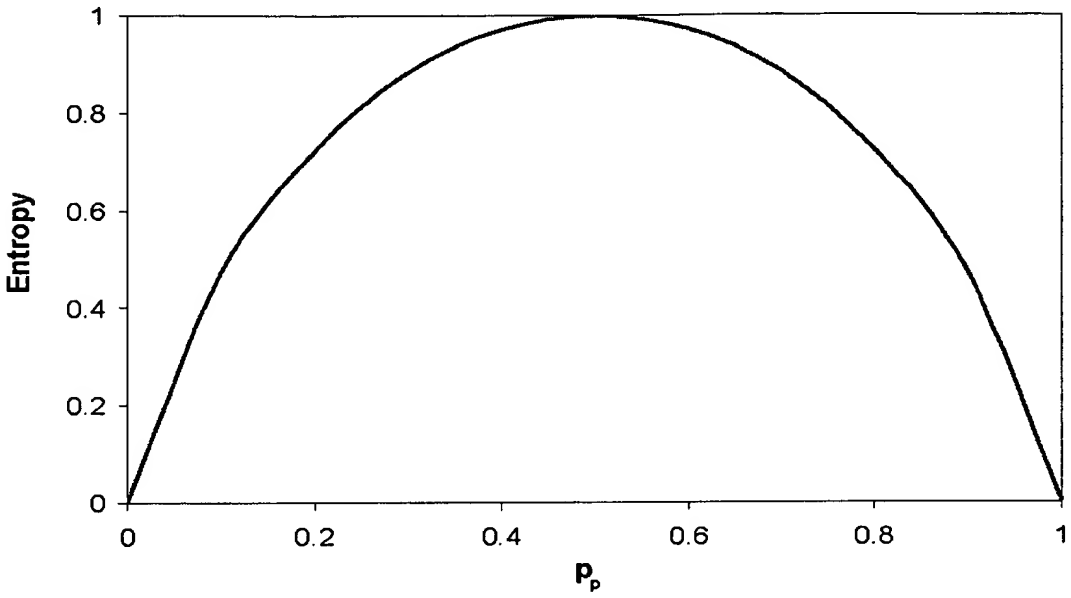

Figure 3: The entropy function relative to a binary classification, as the proportion of positive examples p_p varies between 0 and 1.

One interpretation of entropy from information theory is that it specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S (i.e., a member of S drawn at random with uniform probability). For example, if p_p is 1, the receiver knows the drawn example will be positive, so no message need be sent, and the entropy is 0. On the other hand, if p_p is 0.5, one bit is required to indicate whether the drawn example is positive or negative. If p_p is 0.8, then a collection of messages can be encoded using on average less than 1 bit per message by assigning shorter codes to collections of positive examples and longer codes to less likely negative examples.

Thus far we have discussed entropy in the special case where the target classification is binary. If the target attribute takes on *c* different values, then the entropy of S relative to this *c*-wise classification is defined as

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

where p_i is the proportion of S belonging to class *i*. Note the logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits. Note also that if the target attribute can take on c possible values, the maximum possible entropy is $\log_2 c$.

Information gain measures the expected reduction in entropy

Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. The measure we will use, called *information gain*, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. More precisely, the information gain, *Gain (S, A)* of an attribute *A*, relative to a collection of examples S, is defined as

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where *Values(A)* is the set of all possible values for attribute *A*, and S_v is the subset of *S* for which attribute *A* has value *v* (i.e., $S_v = \{s \in S \mid A(s) = v\}$). Note the first term in the equation for *Gain* is just the entropy of the original collection *S* and the second term is the expected value of the entropy after S is partitioned using attribute *A*. The expected entropy described by this second term is simply the sum of the entropies of each subset S_v, weighted by the fraction of examples $|S_v|/|S|$ that belong to S_v. *Gain (S,A)* is therefore the expected reduction in entropy caused by knowing the value of attribute *A*. Put another way, *Gain(S,A)* is the information provided about the target attribute value, given the value of some other attribute A. The value of *Gain(S,A)* is the number of bits saved when encoding the target value of an arbitrary member of *S*, by knowing the value of attribute *A*.

The process of selecting a new attribute and partitioning the training examples is now repeated for each non-terminal descendant node, this time using only the training examples associated with that node. Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree. This process continues for each new leaf node until either of two conditions is met:

1. every attribute has already been included along this path through the tree, or

2. the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

Issues in data mining with decision trees

Practical issues in learning decision trees include determining how deeply to grow the decision tree, handling continuous attributes, choosing an appropriate attribute selection measure, handling training data with missing attribute values, handing attributes with differing costs, and improving computational efficiency. Below we discuss each of these issues and extensions to the basic ID3 algorithm that address them.

Avoiding over-fitting the data

In principle decision tree algorithm described in Figure 2 can grow each branch of the tree just deeply enough to perfectly classify the training examples. While this is sometimes a reasonable strategy, in fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that *over-fit* the training examples.

Over-fitting is a significant practical difficulty for decision tree learning and many other learning methods. There are several approaches to avoiding over-fitting in decision tree learning. These can be grouped into two classes:

- approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
- approaches that allow the tree to over-fit the data, and then post prune the tree. (emphasis added)

Although the first of these approaches might seem more direct, the second approach of post-pruning over-fit trees has been found to be more successful in practice. This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree. (emphasis added)

Regardless of whether the correct tree size is found by stopping early or by post-pruning, a key question is what criterion is to be used to determine the correct final tree size. Approaches include:

- Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
- Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an

improvement beyond the training set. (emphasis added)

- Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. This approach is based on a heuristic called the Minimum Description Length principle.

The first of the above approaches is the most common and is often referred to as a training and validation set approach. In this approach, the available data are separated into two sets of examples: a *training set*, which is used to form the learned hypothesis, and a separate *validation set*, which is used to evaluate the accuracy of this hypothesis over subsequent data and, in particular, to evaluate the impact of pruning this hypothesis. (emphasis added)

Incorporating Continuous-Valued Attributes

The initial definition of ID3 is restricted to attributes that take on a discrete set of values. First, the target attribute whose value is predicted by the learned tree must be discrete valued. Second, the attributes tested in the decision nodes of the tree must also be discrete valued. This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree. This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals. In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new Boolean attribute A_c that is true if $A < c$ and false otherwise. The only question is how to select the best value for the threshold c. Clearly, we would like to pick a threshold, c, that produces the greatest information gain. By sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of A. It can be shown that the value of c that maximizes information gain must always lie at such a boundary. These candidate thresholds can then be evaluated by computing the information gain associated with each. The information gain can then be computed for each of the candidate attributes, and the best can be selected. This dynamically created Boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree.

Handling Training Examples with Missing Attribute Values

In certain cases, the available data may be missing values for some attributes. For example, in a medical domain in which we wish to predict patient outcome based on various laboratory tests, it may be that the lab test Blood-Test-Result is available only for a subset of the patients. In such cases, it is common to estimate the missing attribute value based on other examples for which this attribute has a known value.

Consider the situation in which *Gain(S, A)* is to be calculated at node n in the decision tree to evaluate whether the attribute *A* is the best attribute to test at this decision node. Suppose that $<x, c(x)>$ is one of the training examples in *S* and that the value *A(x)* is unknown, where *c(x)* is the class label of *x*.

One strategy for dealing with the missing attribute value is to assign it the value that is most common among training examples at node n. Alternatively, we might assign it the most common value among examples at node n that have the classification *c(x)*. The elaborated training example using this estimated value for *A(x)* can then be used directly by the existing decision tree learning algorithm.

A second, more complex procedure is to assign a probability to each of the possible values of *A* rather than simply assigning the most common value to *A(x)*. These probabilities can be estimated again based on the observed frequencies of the various values for *A* among the examples at node n. For example, given a Boolean attribute *A*, if node n contains six known examples with $A = 1$ and four with $A = 0$, then we would say the probability that $A(x) = 1$ is 0.6, and the probability that $A(x) = 0$ is 0.4. A fractional 0.6 of instance x is now distributed down the branch for $A = 1$ and a fractional 0.4 of x down the other tree branch. These fractional examples are used for the purpose of computing information *Gain* and can be further subdivided at subsequent branches of the tree if a second missing attribute value must be tested. This same fractioning of examples can also be applied after learning, to classify new instances whose attribute values are unknown. In this case, the classification of the new instance is simply the most probable classification, computed by summing the weights of the instance fragments classified in different ways at the leaf nodes of the tree. This method for handling missing attribute values is used in C4.5.

Strengths and Weakness of Decision Tree Methods

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
- Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared. (emphasis added)
- Decision trees do not treat well non-rectangular regions. Most decision-tree algorithms only examine a single field at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.

Hence, it is respectfully submitted that the terminology "pruning a retrieval path" in amended claims 5, 21, 25, 26 and 27 is definite and claims 5, 21, 25, 26 and 27 are in allowable form under 35 U.S.C. §112, second paragraph.

For clarity, claim 5 has been amended to recite, in part:

> generating, by a superposition calculating unit, a combination of correspondences comprising generating a combination by generating a decision tree having at least one retrieval path, the decision tree being based on minimized root mean square distance values between the first and second point sets, setting a predetermined threshold value and pruning a retrieval path if an attribute value of a candidate for the combination of correspondences is greater than the predetermined threshold value, and determining that a point is a candidate if an attribute of an element of the first point set includes a type of an atom, an atomic group, a molecule, a hydrophilic property, a hydrophobic property, or a positive or negative charge that coincides with an attribute of an element of the first point set, and refining the elements of the first and second point sets based on an attribute of the elements of the first and second point sets to provide a first subset and a second subset, respectively.

The insertion of the terminology "for the combination of correspondence," which is based on, for example, page 8, lines 28-33, is believed to clarify the terminology "candidate." Claims 21, 25, 26 and 27 have been amended correspondingly.

Hence, the terminology "candidate" is now submitted to be definite and claims 5, 21, 25, 26 and 27 are submitted to be definite and in allowable form in allowable form under 35 U.S.C. §112, second paragraph.

Independent claim 5 been amended to correct "first" to recite ---second--- as follows: "determining that a point is ~~a~~ the candidate if an attribute of an element of the first point set includes a type of an atom, an atomic group, a molecule, a hydrophilic property, a hydrophobic property, or a positive or negative charge that coincides with an attribute of an element of the ~~first~~ second point set." Independent claims 21, 25, 26, and 27 have been amended correspondingly. Applicants thank the Examiner for pointing out this typographical error.

Hence, amended independent claims 5, 21, 25, 26 and 27 are submitted to be definite and allowable under 35 U.S.C. §112, second paragraph with respect to the change of "first" to ---second--- as described above.

The Examiner requests clarification of the terminology regarding refining elements of point sets "based on" an attribute of elements of the two point sets, and requests parameters that are required and/or applied. It is respectfully submitted that the attributes are defined on page 36, line 36 through page 37, line 10, of the specification:

> (6) Refining of candidates based on an attribute of a point
> The number of candidates for a point to be related can be reduced by using an attribute of the point in relating an element a_i of a point set A to an element b_1 of a point set B. The attributes of the point, for example, include the type of an atom, an atomic group, and a molecule, the hydrophilic property, the hydrophobic property, and the positive or negative charge. It is determined whether the point is selected as a candidate by checking whether these attributes coincide.

That is, in refining, it is determined whether the point is selected as a candidate by checking whether these attributes coincide. Independent claim 5 has been amended to recite: "based on ~~an~~ coinciding attributes of the elements of the first and second point sets." Independent claims 21, 25, and 26 have been amended in corresponding fashion. Hence, it is respectfully submitted that amended claims 5, 21, 25, and 26 are definite and in allowable form under 35 U.S.C. §112, second paragraph.

The Examiner submits that it is unclear in the determining...based on... and minimizing rmsd... if the applicants intend an active method step of minimizing rmsd or intends an active method step of minimizing rmsd or intends determining (similar portions) based on minimized rmsd. On page 28, lines 5-13, the specification recites:

> For instance, it is assumed that there are substances expressed by a point set A = $\{a_1, a_2, ..., a_i, ..., a_m\}$ as shown in Fig. 13A and a point set B = $\{b_1, b_2, ..., b_j, ..., b_n\}$ as shown in Fig. 13B. The elements constituting these substances A and B are related to each other as shown in Fig. 13C, and the substance B is rotated and moved so that the r.m.s.d value between the corresponding elements is minimized, as shown in Fig. 13D. (emphasis added)

Hence, it is respectfully submitted that the superposition calculating unit rotates the second point set based on the generated combination of correspondence so that the r.m.s.d value between the corresponding elements is minimized. Thus, it is respectfully submitted that one skilled in the art understands the terminology "determining...based on... and minimizing rmsd..." of claims 5, 21, and 25. Hence, claims 5, 21 and 25 are submitted to be definite and in allowable form under 35 U.S.C. §112, second paragraph, with respect to the terminology "determining...based on... and minimizing rmsd...."

Claims 6-7, 9, 11 and 24 have been amended to correct antecedent basis and are now submitted to be definite and in allowable form under 35 U.S.C. §112, second paragraph. Claims 8 and 10 have been cancelled without prejudice or disclaimer.

Claim 21 has been amended to correct antecedent basis and is now submitted to be definite and in allowable form under 35 U.S.C. §112, second paragraph.

Since claims 6-11 and 24 depend, directly or indirectly, from amended claim 5, claims 6-11 and 24 are submitted to be definite and in allowable form under 35 U.S.C. §112, second paragraph for all the above reasons cited for amended claim 5.

CONCLUSION:

In accordance with the foregoing, it is respectfully submitted that all outstanding objections and rejections have been overcome and/or rendered moot. And further, that all pending claims patentably distinguish over the prior art. Thus, there being no further outstanding objections or rejections, the application is submitted as being in condition for allowance which action is earnestly solicited.

If the Examiner has any remaining issues to be addressed, it is believed that prosecution can be expedited by the Examiner contacting the undersigned attorney for a telephone interview to discuss resolution of such issues.

If there are any underpayments or overpayments of fees associated with the filing of this Amendment, please charge and/or credit the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: August 2, 2006 By: Darleen J. Stockley
Darleen J. Stockley
Registration No. 34,257

1201 New York Avenue, N.W.
Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501